

Remarks

Entry of the amendments, reconsideration of this application in light of the remarks below and allowance of all pending claims are respectfully requested. Claims 45 - 64 remain pending in the application.

The foregoing changes have been made by applicant in the claims to better define the present invention. The phrase "said program being absent embedded debug commands" is not deemed to be necessary, and therefore, moved to a dependent claim.

Support for this amendment is explicitly shown in the source code described in applicant's specification, which has no embedded debug commands (see, e.g., paragraphs 20 and 21 of applicant's specification). Therefore, no new matter is added. This is more fully described below.

Substitute drawings have been sent in order to correct an omission in FIG. 1A thereof. More specifically, FIG. 1A has been amended to insert reference numerals "116" therein to identify the two network interface cards depicted in this figure at the respective ends of connection 104. The network interface cards are specifically identified in paragraph 00022 of the specification as filed so adding the missing reference numerals, as described above, will not involve the addition of new matter.

35 U.S.C 112, First Paragraph, Claim Rejection

In paragraph 3 of the Office Action, dated July 12, 2006, claims 45 - 64 have been rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement thereof. Independent claims 45, 53 and 59 are rejected on the basis that they contain subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Specifically, the Office Action objects to the limitation "said program being absent embedded debug commands." The Office Action further states that Applicant's specification

as originally filed does not include any description of “embedded debug commands,” and accordingly cannot provide the necessary support for this limitation.

Applicant respectfully submits that this rejection has been interposed in error. The claim limitation referred to in the Office Action, which is now in dependent claim 52, instead of independent claims 45, 53 and 59, recites an absence of embedded debug commands. Applicant’s specification as originally filed made no express reference to and did not include embedded debug commands.

An explanation of and foundation for inclusion of the "said program being absent embedded debug commands" limitation was set forth in Applicant’s Amendment dated December 14, 2004 by which the claims were amended to include the limitation “said program being absent embedded debug commands.” That explanation is repeated here for convenience of the examiner.

“Applicants have amended the originally filed independent claims to further define the environment of one aspect of the invention. For example, the originally filed independent claims have been amended to indicate that the program which is being debugged does not include embedded debug commands. Support for this amendment is provided throughout applicants’ specification. For example, the exemplary source code depicted on pages 20 and 21 are devoid of debug commands. That is, applicants’ specification and the examples provided therein depict source code with no debug commands, providing support that the program does not include embedded debug commands. Thus, no new matter is added.”

At a later point in that same December 14, 2004 response, applicant notes, “The program being debugged does not include embedded debug commands. Instead, breakpoints are provided by a debugger during a debug session.”

Accordingly, applicant respectfully submits that the claims as amended, particularly dependent claim 52, are in full compliance with and met their statutory obligations under 35 U.S.C. 112, first paragraph, in this instance. Applicant further respectfully submits that the addition of the limitation "said program being absent embedded debug commands" is supported by and reflected in the specification as originally filed.

Basis of Individual Arguments vs. References

Applicant respectfully submits that the combination of references has been attacked, and not just the individual references. The individual references were discussed to show that each reference is missing at least one of the same claimed elements and, therefore, the combination is also missing that element. Applicant discuss the individual references initially for clarity, but then fully discusses why the combination does not teach or suggest applicant's claimed invention.

Applicant respectfully submits that this approach is warranted in this instance and will address the teachings of the prior art, singly and in combination, after the following discussion of the 35 U.S.C. 112, first paragraph, rejection.

35 U.S.C. 103(a) Claim Rejection

Claims 45–64 have been rejected under 35 U.S.C. 103(a) as being unpatentable over U.S. Patent No. 6,263,489 to Olsen et al (“Olsen”) in view of U.S. Patent No. 6,282, 698 to Baker et al (“Baker”). Applicant again respectfully traverses this rejection for the reasons previously given and as set forth below.

One aspect of applicant's invention is directed to automatically restoring debugging breakpoints after modifying the source code of a program to correct errors discovered therein. The automatically restoring feature restores the breakpoint to the same step the breakpoint was set to prior to modification of the source code, although that step is now at a different location within the modified program.

As an example, to automatically restore the debugging breakpoint to the same step of the program that had the breakpoint prior to modification, the location (e.g., line of code) that includes the selected step is determined. This determination is made by creating an instruction profile that includes attributes of a number of generated instructions that are associated with the selected step. These attributes include attributes of the instruction (such as, an operation code, operand and operation type obtained from reading the instructions) of the selected step, and possibly attributes of other instructions in proximity to the selected

step. It is important to note that this element is expressly included in each of applicant's independent claims and that it is not taught by the individual references or their combination.

When the program is modified, the attributes of the instruction profile are compared to the attributes of the newly generated instructions to determine where the particular line of code of interest has moved in the program. The various comparisons made to different instructions within the modified program yield difference counters. In this example, the difference counter having the smallest value indicates the location of the selected step. Thus, the debugger automatically restores the breakpoint to that step. A difference counter is not taught in or suggested by the references.

In one particular example, applicant claims a method of restoring debugging breakpoints (e.g., independent claim 45). The method includes, for instance, having a breakpoint that is set to a selected step of a first version of source code of a program, the program being absent embedded debug commands; creating an instruction profile for the selected step, the instruction profile including one or more attributes of one or more machine instructions generated for the selected step and one or more attributes of zero or more other machine instructions generated for the first version of source code; and automatically restoring the breakpoint to the selected step of a modified program, in response to modification of the first version of source code to provide the modified program having a second version of source code, wherein the selected step is at a different location within the modified program, and wherein the automatically restoring includes comparing one or more attributes of the one or more machine instructions generated for the second version of source code with one or more attributes of the instruction profile created based on the first version of source code to determine the different location. These features are very different from the teachings of Olsen and Baker, either alone or in combination.

In the *Response to Arguments* section of the July 12, 2006 Office Action, Olsen is described as teaching a method for automatically restoring a breakpoint (see, for example, column 13, lines 8-12) to a selected step of a modified program (see, for example, column 5, lines 28-30), as does applicant. However, this portion of the Olsen description only teaches that a breakpoint instruction is substituted for the machine code instruction positioned at the

HW and the machine code instruction at that position is stored in a table for later use. The attributes of the machine code instruction are not considered or used in any manner to form a profile or for subsequent comparisons and no such teaching appears in or is suggested by Olsen.

Applicant respectfully and further submits that this teaching in Olsen is directed to insertion of a breakpoint instruction into binary, not source code, is not automatic and does not use any characteristics of the replaced machine code instruction to determine the selected step. In fact, the table in Olsen that is relied on as a basis for asserting that Olsen creates an instruction profile is nothing more than a means of associating the inserted breakpoint instruction with the entire machine code instruction it replaced rather than any of the attributes of that instruction. This is merely a swap of information on an “as is” basis and not a profile determined by machine instruction attributes as is expressly specified in all of applicant’s claims.

Per the Office Action, the Olsen breakpoint is initially set to a selected step P in the source code of the program prior to modification of the program after which the selected step will be at a different location within the program once it is modified. Applicant agrees that Olsen shows insertion of a source code breakpoint that needs to be subsequently located after program modification, but that one step falls far short of teaching the remainder of applicant’s invention as claimed.

It is stated in the Office Action that Olsen further teaches that restoring the breakpoint comprises comparing one or more attributes of one or more machine instructions generated for the source code with one or more attributes of an instruction profile created based on the source code (see, for example, column 12, lines 15-63) to determine the different location of the selected step (see, for example, column 12, line 64 to column 13, line 4). Again, this reference to teachings in Olsen fails to define or teach the creation and use of an instruction profile as taught and claimed by applicant as is now explained.

Moreover, it is important to note that in Olsen the restored breakpoint is inserted back into the same modified program where it was originally inserted. This is clearly not the case with applicant’s invention where the instruction profile is based on a code instruction or more

associated the first version of the source code and the breakpoint is restored into a source version of the modified or debugged program. Although Baker is discussed at a later point in this response, it is important to note here that no such breakpoint insertion takes place in Baker at any time.

The remainder of the column 12, lines 15 – 63, description is devoted to the steps used by Olsen to calculate the start and end points of a block of instructions of interest to a user. The attributes of the machine instructions captured in the block or blocks of interest are not considered or saved and no profile comparable to that of applicant is, therefore, created or referenced in attempting to restore a breakpoint after optimization in Olsen occurs.

The Office Action next notes that the source code Olsen discloses is considered a "first version" of source code, although Olsen doesn't disclose or suggest how its teachings could be used if another source code version of the program was available. Applicant agrees with this conclusion as to the lack of second program version disclosure, but wishes to point out that Olsen also fails to teach other important aspects of applicant's claimed invention, such as the creation and use of an instruction profile based on machine instructions directly associated with a breakpoint set in a first version of a program and automatic restoration of that breakpoint in a second version of the program based on the profile so created. All such requirements are expressly set forth in and form part of the pending claims.

In particular, Olsen fails to describe, teach or suggest applicant's claimed element of comparing one or more attributes of one or more machine instructions generated for the second version of the source code with one or more attributes of the instruction profile created based on a first version of the source code to determine the location in which to restore the breakpoint after modification of the source code. The failure of Olsen to teach or suggest this aspect of applicant's claimed invention is explicitly admitted in the January 13, 2006 Office Action. Although Baker is relied upon to close this gap, Baker, fails to overcome the deficiencies of Olsen for the reasons set forth below.

The Office Action argues that one of ordinary skill in the art would have been motivated to automatically restore a breakpoint to a selected step of a modified program, regardless of whether it is an optimization (as in Olsen), a modification of the source code (as

in Baker), or some other operation that provides the modified program. That argument is apparently necessary because the hypothetical combination urged as obviating applicant's claimed invention lacks this necessary step. The motivation to automatically restore a breakpoint using applicant's instruction profile seems to have come from applicant, not the prior art. This point is discussed and expanded on below.

With respect to the role played by one of ordinary skill in the art, the Examiner's attention is respectfully directed to the holding in *Ex parte Levengood*, 28 USPQ 2d 1300 (Bd. Pat. App. & Inter. 1993). *Levengood* held that the fact one of ordinary skill in the art has the capabilities to arrive at the invention is not the test for whether one of ordinary skill in the art would have arrived at the invention based on the teachings of the prior art; "That which is within the capability of one skilled in the art, is not synonymous with obviousness." Applicant respectfully submits that *Levengood* applies in all respects to the asserted basis of the current rejection.

In support of this proposition, Baker is cited for a teaching that a first version of source code can be modified to provide a second version of the program (see, for example, column 1, line 64 to column 2, line 8), and thereafter, compared to determine their similarity by comparing attributes of one or more binary code instructions generated for the different versions of the source code, so as to find similarities in the programs even when the source code is not accessible (see, for example, column 3, lines 17-31). The binary bytecode instructions are disassembled and preprocessed to create an instruction profile that comprises one or more attributes of the instructions (see, for example, blocks 120 and 130 in FIG. 1).

Applicant respectfully submits that this view of Baker is overly generous, if not erroneous, and attributes a teaching to Baker not actually found in or contemplated by that reference. What takes place in blocks 120 and 130 is merely the analysis and processing of disassembled bytecodes (not binary code instruction attributes as in applicant's invention), the computation of offsets in a manner that negates code comparison mismatches and the further breakdown of the disassembled bytecodes into tokens for subsequent comparison of the two programs, not to locate a single breakpoint instruction in a modified program.

Baker is directed at a technique for detecting similarities in large sets of binary code files without requiring access or knowledge of the actual source code itself. In Baker, these binary code file sets are disassembled and transformed for analysis by similarity text, not binary, detection tools. Baker does not teach isolating and then comparing attributes of one or more binary code instructions to find a specific instruction or its location in the original version of the target program. Rather, in Baker, comparison of the binary files is made in gross, on a bulk basis, with no focus on or effort made to determine the identity, location or attributes of a specified instruction or breakpoint location in either the original or modified programs described in Baker.

The program similarities the Baker technique seeks are used solely for purposes of comparing large sets of binary code to determine if a user is employing the latest program in a sequence of program versions that have been upgraded or enhanced over time. Baker is not directed at debugging programs, restoring debugging breakpoints or relocating them in accordance with machine instruction profiles, contrary to the cooperation between and use of elements by which such capability is claimed by applicant. Baker does not suggest or teach any focus on or concern with individual source code lines or instructions associated with specific source code. Lastly, there is no profile created in Baker for a single or a small set of instructions. Instead, the bytecodes that form the Baker program versions are disassembled, encoded and preprocessed before the text based comparison tools are applied; see column 3, lines 16 – 30 of Baker. An instruction profile as taught and employed by applicant is, therefore, not suggested or even possible given this approach. As a result, even if Baker were combined with Olsen, the resulting combination would still not identify the location of a specific source code line or breakpoint in the modified form of program being debugged as the result of comparing it to the original source code version thereof.

The Office Action suggests that Baker's instruction profile enables the desired comparison even when the different versions of source code are not accessible, thus providing more opportunities to apply Olsen's teachings. However, as noted above, source code is, in fact, created in Baker through disassembly although no attention is paid in Baker to source code line location or instructions associated with source code lines since that information is not sought nor relied on. Baker is directed at determining how many instances

of certain code attributes there are on a total basis in each of the two program versions being compared and in then making a determination of similarity based on the gross results of such comparison. A person having ordinary skill in this art, having Olsen in hand, would not be inclined to seek out Baker since it offers no teaching or assistance in changing Olsen to match applicant's claimed invention.

The Office Action relies on the combination of Baker's teaching of modifying a first version of source code to provide a modified program having a second version of source code in view of Olsen's teaching of automatically restoring a breakpoint to a selected step of a modified program concluding from this supposition that it would have been obvious to one of ordinary skill in the art at the time the invention was made to automatically restore a breakpoint to a selected step of a modified program having a second version of source code.

Applicant contends that this conclusion is based on a generalization of the Baker and Olsen teachings and that, in any event, the Olsen-Baker combination stops well short of the combination of elements claimed by applicant. As previously noted, Olsen and Baker are each trying to solve significantly different problems than applicant and are not reasonably pertinent to the particular problem that applicant's invention solves or to the express manner in which the invention is claimed.

According to the Office Action, applicant alleges generally that the combination or suggested modification appears to be a hindsight reconstruction of Applicant's invention. However, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

Applicant respectfully submits, there is nothing in Olsen or Baker or their combination as set forth in the Office Action that suggests use of an instruction profile based on attributes of machine instructions as called for by applicant in the present claims or the automatic restoration of a breakpoint predicated specifically on applicant's instruction

profile. Applicant respectfully submits that this is an instance in which the gap between that missing from the references, as described above, is supplied by a suggestion from the claimed invention rather than the reverse. In *Iron Grip Barbell Company, Inc. v. York Barbell Company, Inc.*, 392 F.3d 1317, 73 USPQ2d 1225, 1227 (Fed. Cir. 2004), the court said, “We turn first to a comparison between the prior art and the claimed invention. In this inquiry, we are mindful of the repeated warnings of the Supreme Court and this court as to the danger of hindsight bias. See, e.g., *Graham*, 383 U.S. at 36 (consideration of secondary factors “serve[s] to guard against slipping into use of hindsight and to resist the temptation to read into the prior art the teachings of the invention in issue” (internal quotations and citations omitted)); *In re Kotzab*, 217 F.3d 1365, 1369 (Fed. Cir. 2000) (“[T]he very ease with which the invention can be understood may prompt one to fall victim to the insidious effect of a hindsight syndrome wherein that which only the invention taught is used against its teacher.” (internal quotations omitted)). We note in this respect that the district court’s use of an “overall picture” and “common sense” test of obviousness falls squarely into the hindsight trap. See *In re Lee*, 277 F.3d 1338, 1345 (Fed. Cir. 2002).”

“Where an invention is contended to be obvious based upon a combination of elements across different references,” as is the situation in this instance, “our cases require that there be a suggestion, motivation or teaching to those skilled in the art to hypothecate that combination, *In re Fine*, 837 F.2d 1071, 1074 (Fed. Cir. 1988). This requirement prevents the use of “the inventor’s disclosure as a blueprint for piecing together the prior art to defeat patentability—the essence of hindsight.” *Ecolochem, Inc. v. So. Cal. Edison Co.*, 227 F.3d 1361, 1371-72 (Fed. Cir. 2000) (quoting *In re Dembiczak*, 175 F.3d 994, 999 (Fed. Cir. 1999)).

Applicant wishes to respectfully point out that there has been no suggestion, motivation or teaching for those skilled in the art to combine Olsen and Baker in the manner used in the Office Action. Olsen is directed at a situation involving optimized code while Baker is concentrated exclusively on comparing large sets of code, if not entire programs, to see if they are different. Accurately relocating breakpoints and the absence in either reference of an instruction profile used for this purpose as expressly called for in applicant’s

pending claims patentably separates applicants invention from the hypothetical combination proposed in the Office Action.

Applicant expressly claims an inventive method, system and article of manufacture by which or wherein a debugging breakpoint is automatically restored to the same step or point of the program that incorporated the breakpoint prior to debugging of the program and its resultant modification. The determination of the post debugging location to which the breakpoint is restored is made by creating and thereafter utilizing an instruction profile that includes attributes of a number of generated machine instructions that are originally associated with the selected step. These include attributes of the instructions (such as, an operation code, operand and operation type obtained from reading the instructions) of the selected step, and possibly attributes of other instructions in original proximity to the selected step, none of which is suggested in or taught by the references. This means that the combination formed from the references is also lacking such inventive elements.

As claimed, when the program is modified, the attributes of the instruction profile are compared to the attributes of newly generated instructions in the modified program to determine where the particular line of code of interest has been moved. The various comparisons made to different instructions within the modified program yield difference counters. In this example, the difference counter having the smallest value indicates the location of the selected step. Thus, the debugger automatically restores the breakpoint to that step, a claimed feature not found in either of the references or their combination.

The combination of the Olsen and Baker references (even assuming *arguendo* that the combination is proper) merely teaches that a determination may be made as to whether two binaries are similar and that a breakpoint can be restored in optimized code. There is no teaching or suggestion in the combination of automatically restoring a breakpoint in a modified program having a second version of source code, as claimed by applicant. In particular, there is no teaching or suggestion in the combination of automatically restoring the breakpoint to a selected step of a modified program, in response to modification of the first version of source code to provide the modified program having a second version of

source code. The determination that two binaries are similar is not a teaching of what is claimed by applicant.

Baker does not help Olsen address the problem applicant has solved. Applicant by definition already knew that the programs were similar – one was a modified version of the other. What applicant was trying to solve was the problem of automatically restoring a breakpoint to a selected step that was in a different location in a second version of source code. This is not taught by Baker, which simply provides a way to determine whether two binaries are similar, something applicant already knew, without use of applicant's instruction profile. Further, it is not taught by Olsen, which describes restoring optimized code, also without use of anything like applicant's instruction profile. There is no teaching in the combined references of how one would automatically restore a breakpoint to a selected step of a modified program having a second version of source code. Thus, applicant respectfully submits that the combination fails to describe, teach or suggest applicant's claimed invention.

According to *Ex parte Clapp*, 227 USPQ 972 (Bd. Pat. App. & Inter. 1985), the Office Action has the initial obligation to show that the prior art combination provides a suggestion of doing what the inventor has done. "To support the conclusion that the claimed invention is directed to obvious subject matter," *Ex parte Clapp* at 973, "either the references themselves must expressly or impliedly suggest the claimed invention or the examiner must present a convincing line of reasoning as to why the artisan would have found the claimed invention to have been obvious in light of the teachings of the references." It is respectfully submitted that this test has not been met in this instance.

Further, obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention in accordance with a teaching, suggestion, or motivation to make that hypothetical combination. "The test for an implicit showing is what the combined teachings, knowledge of one of ordinary skill in the art, and the nature of the problem to be solved as a whole would have suggested to those of ordinary skill in the art." *In re Kotzab*, 217 F.3d 1365, 1370, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000), *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988). The fact that two or more common or complimentary subject matter references can be combined does not in and of

itself render the resultant combination obvious unless the prior art also suggests the desirability of the combination. *In re Mills*, 916 F.2d 680, 16 USPQ2d 1430 (Fed. Cir. 1990)

A conclusory statement that a combination of prior art references obviate the claimed invention because the combination would have been " 'well within the ordinary skill of the art at the time the claimed invention was made' " is insufficient to establish a *prima facie* case of obviousness without some objective reason to combine the teachings of the references. *Ex parte Levengood*, supra; see also *In re Kotzab*, supra.

Most importantly, to establish *prima facie* obviousness of a claimed invention, each of the claim limitations must be taught or suggested by the prior art, *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). This has not been done in this instance.

For the foregoing reasons, applicant respectfully submits that the combination of Olsen and Baker fails to describe, teach or suggest one or more aspects of applicant's claimed invention. Additionally, applicant respectfully submits that the combination of Olsen and Baker is improper in that one skilled in the art would not look to Baker, a reference concerned only with comparing large sets of binary code or entire programs, to supply what was missing in Olsen.

Lastly, based on the prior cited holdings and for the reasons given above, it is respectfully submitted that it is error to extract a suggestion to combine references from the invention under review. Applicant respectfully submits that the combination of Olsen and Baker is improper in that respect. In addition, applicant submits that since there is no teaching or suggestion in the references themselves to make the combination or modification suggested in the Office Action, the combination itself is without merit. It is well known that:

It is insufficient to establish obviousness that the separate elements of the invention existed in the prior art; absent some teaching or suggestion, in the prior art to combine the elements. *Arkie Loures Inc. v. Gene Larew Tackle Inc.*, 43 U.S.P.Q. 2d 1294, 1297 (Fed. Cir. 1997).

Applicant respectfully submits that there is no such teaching or suggestion in the references. All prior discussion of the combination advanced in the Office Action fail to

indicate where the references expressly teach or suggest the combination. Rather, as previously pointed out, the combination or suggested modification to Olsen appears to merely be a hindsight reconstruction of applicant's invention. That is, there is no basis or justification for selecting various elements of Olsen and Baker to form a combination that would not have been obvious to one of ordinary skill in the art at the time the invention was made.

This lack of suggestion or knowledge is particularly important since Olsen and Baker are each trying to solve different problems, and are not reasonably pertinent to the particular problem with which the inventor is involved. Baker has no relation to Olsen. Olsen is concerned with setting breakpoints in an optimized version of source code. In Olsen, there is only one version of source code. There is no discussion at all in Olsen of wanting or needing multiple versions of the source code. Olsen is merely concerned with debugging optimized code of one version of source code. In contrast, Baker is concerned with multiple versions of code and not at all with debugging the code or setting breakpoints. Baker is directed to determining whether two binaries are similar. Thus, Olsen and Baker are directed at different problems and would not be combined by one of ordinary skill. Neither reference is directed to the problem applicant solved, which solution includes determining how to automatically restore a breakpoint of a modified program having a second version of source code. Baker is not even concerned with debugging at all.

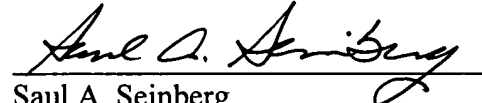
Additionally, applicant respectfully submits that all of the dependent claims (46 – 52, 54 – 58 and 60 – 64) are patentable for the same reasons as the independent claims, as well as for their own additional features. If an independent claim is nonobvious under 35 U.S.C. 103, then any claim depending therefrom is nonobvious, *In re Fine*, supra.

For instance, applicant explicitly recites in dependent claims 47, 55 and 61 that the instruction profile includes a source line number for the selected step and the length of the first version of source code and that the automatically restoring uses these values to determine a starting point within the modified program to select the one or more instructions generated for the second version to be used in the comparing. This is not described, taught or suggested in Olsen or Baker, either alone or in combination.

For all of the above reasons, applicant respectfully submits that all claims currently pending herein are patentable over the combination of Olsen and Baker and any other cited art.

Should the examiner wish to discuss this case with applicant's attorney, please contact applicant's attorney at the below listed number.

Respectfully submitted,

A handwritten signature in cursive script, appearing to read "Saul A. Seinberg", is written over a horizontal line.

Saul A. Seinberg
Attorney for Applicant
Registration No.: 24,840

Dated: October ____, 2006.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579